NeoSOFT – Web Project Guidelines

For

MS DOT NET Department

V1.1

Document Revision History

Ver. No.	Date	Change Information	Author	Reviewed/ Approved By	Reviewed/ Approved Date	Release Date
1.0	22nd Sept,2016	Initial Draft	Vaibhav Kulkarni	Bhavesh Patel	22nd Sept,2016	
1.1		Consolidated Draft	Vaibhav Kulkarni	Bhavesh Patel		

Template Revision History

Ver. No.	Date	Change Information	Author	Reviewed/ Approved By	Reviewed/ Approved Date	Release Date
1.0	22nd Sept,2016		Vaibhav Kulkarni	Bhavesh Patel	22nd Sept,2016	

Table of Contents

Website Level Settings

1. Each page should have a clear Title specified.
2. Site should have a proper favicon image.
3. Every site must have a logo image.
4. Logo should point to home page of the site.

   Note: Pre-login and Post-login home/landing pages will be different, so accurate linking is important.

5. Site pages must follow a uniform layouts as per design.
6. Each image in the website must have an **'alt'** attribute
7. Use of “No Image” image is mandatory if actual image is not present.
8. Use of Ajax Loader is mandatory while loading any heavy objects like Grid/List or any heavy page so that user will be aware and page will not become irresponsive.

HTML Guidelines

1. HTML tags should be in lower case.
   Not Good : <DIV>,
   Good: ,
2. Every opening tag must have closing tag.
   Not Good : , <hr>,
   Good: ,<hr/>,
3. HTML Element ID attribute should follow Pascal Casing. CSS class name should be in lower case.
   Not Good:
   Good:
4. Use H1 – H6 tags. For semantic and SEO reasons, try to use heading tags. The hierarchy of them in the page is important too.

5. Place a comment tag at the end of each important HTML container that notifies the end of that specific tag

Jquery/JavaScript Guidelines

1. Wherever possible, Jquery files should be used from Google CDN.
2. In case of ASP.NET MVC, bundling must be used wherever possible.
3. Wherever possible, Jquery files should be in minified version (example – min.js)
4. All custom Jquery/JavaScript code should be placed in external files instead of Form or View.
5. Custom JS files must follow versioning. A latest version should be referenced so that it will prevent caching.
6. Jquery files should be loaded first before any custom JS.
7. Custom JS should be bundled wherever possible.
8. For accessing multiple DOM elements CSS class should be used as it allows reusability across multiple functions.
9. For single DOM element access in Jquery, ID selector should be used as it derives document.getElementById() internally from JavaScript.
10. Jquery inbuilt functions are recommended to use instead of writing custom JavaScript equivalent for the same.
11. Use “debugger” instead of Alert Box during development and unit testing.
12. All debuggers and alert box, if any, must be disabled/removed before publishing the code for deployment.
13. Always use proper “HTTPGET, HTTPPOST, HTTPDELETE, HTTPPUT” Ajax request as per CRUD operations. (Example : GetUser (HTTPGET), InsertUser (HTTPPOST)).
14. While using any Jquery UI plugin, browser compatibility should be checked before.

CSS Guidelines

1. Use of inline CSS should be avoided.
2. CSS should be mentioned in separate files.
3. All CSS files should be loaded at the start of the page in the <head> section, before loading any JS file.
4. CSS classes have to be reusable.
5. For any custom CSS class, start and end section comments should be present.
6. While using any third party design templates or bootstrap templates, sequence of CSS and JS provided by template, should be followed as it is.

Forms Development

1. Every form should have proper heading which specify the purpose of Form.
2. All Form fields must be properly aligned as per designs.
3. Each field in the form should have a proper label associated with it.
4. Label should be very clear and understandable.
5. Label should follow Pascal Casing with a white space in between the words.

   Not Good: FirstName

   Good: First Name

6. **Form Validations**
   a. Every input form field must have proper validations applied.
   b. Use of both, client side and server side validations, is mandatory.
   c. Each input field must be limited with max character limits.
   d. For all field validations, a note should be given "**All * marked fields are mandatory**" on top of Form.
   e. An * must be suffixed to label for each mandatory field in the form.

      Not good: Label should not be *First Name

      Good: Label should be First Name *

   f. Validation messages should be explanatory and must follow proper naming conventions

      Not good: "email required"

      Good: "An Email Id is mandatory"

   g. Validation messages should be given a proper spacing and should not affect design on any browser.
   h. **Email Fields** – Use of regular expression and if required, remote validations should be applied.
   i. **Date Fields** – Each date field should be validated for valid date format.

i. In case of Start Date and End Date pattern, validation must be present to check that Start Date is less than End Date.

j. **Text Area** – It is recommended to specify character limit and counter for the same while using Text Area. Always set **resize** to **None** unless it is required.

k. If form is for posting any crucial user/payment related data then use of CAPTCHA is mandatory. (Decision has to be taken based on business requirement.)

l. On new page load, older validation messages should not reflect.

7. Dropdown/Combo box width should be adequate to read the internal content.
8. Every dropdown list should have default **– Select One –** element as selected and optionally **– All –** element.
9. Unwanted page post backs should be avoided using Ajax in case of any custom validations.
10. Tab indexes should be maintained properly
11. On form load, a focus must come on first text box in the form.
12. Date picker should show today's day highlighted by default.
13. Whenever rendering any date, a standard **dd-MMM-yyyy** format should be followed.
14. **File Upload functionality**

a. File uploads should have proper file type and size validations.

b. A note should be present about maximum permissible file size and type of file to be uploaded.

c. There should be a single **"Upload"** folder present in the main site.

d. If different types of files are to be uploaded, relevant folders can be created under Upload folder, not outside.

e. Remember to ask Administrator to assign read and write permission to **Upload** folder.

Authentication and Authorization

1. Use of Forms Authentication is recommended in any application.
2. Windows authentication can be used only if such business requirement arises.
3. Asp.Net Membership can be used optionally based on business requirement.
4. For secured application, password should be hashed using Ajax call before posting the form.
5. A forms authentication ticket should be generated which maintains authentication cookie.
6. An expiration period must be set for cookie.
7. **Authentication in Asp.Net Web Forms**
    a. Define a separate Serializable user entity (class) and populate the same as a part of Forms Authentication Ticket.
    b. A forms authentication ticket must be validated on Page Load event of Master page or Page load event of normal page in case master page is not present.
    c. In case ticket is invalid, user should be redirected to login page.

8. **Authorization in Asp.Net Web Forms**
    a. In the same Serializable user entity object, we should store the pages/modules for which the current logged in user has access to.
    b. Create a separate class which is inherited by C#'s **MasterPage** class. Inside that we can check the URLs with the pages/modules which we have kept in Serializable entity.
    c. We need to use this newly created class as a base class for all our content pages, in order to invoke the above authorization process.
    d. In case of visited URL does not match with pages/modules, we should redirect to Custom page which will show a message as "**User is not authorized to access the request page. Please contract administrator**."

9. **Authentication in Asp.Net MVC**
    a. Use of Forms Authentication is recommended.
    b. We need to set below section in **system.web** section in web.config file.

```
<authentication mode="Forms">
        <forms loginUrl="~/Account/Login" timeout="2880" />
</authentication>
```

c. In login post action, we need to fetch required data for the authenticated user in user entity and populate forms authentication cookie with serialized user entity.
d. In case of MVC5, an Authentication Filter can be used.
e. To use authentication filters, we need to create Custom Authentication class which will inherit **FilterAttribute** class and implement **IAuthenticationFilter** interface.
f. We need to override **OnAuthentication** method to check whether user is authenticated or not.
    i. De-serialize cookie and retrieve user entity in this event do as to check the user is authenticated or not.
g. In case lower MVC versions, we should use forms authentication only.
h. However, authentication can be combined with Authorization process itself. Please see Authorization In Asp.Net MVC section for the same.

10. Authorization in Asp.Net MVC

a. In the same Serializable user entity, we should store the pages/modules for which the current logged in user has access to.
b. In the same **OnAuthorization** method, we should de-serialize the cookie value to check the page and module access.
c. To invoke this custom authorization, we should use the newly Custom Authorize attribute over our controllers or actions, depending on the requirement.
d. In the custom authorize attribute, we should check if the cookie is present or not. If the cookie does not exists, we should redirect user to login page.
e. If cookies exists, we should check whether the current URL has been given access to user or not. If not, we should redirect user to UnAuthorize page.
f. In case of Ajax requests, we should check whether the request is AJAX request or not. We can check the same as below,

```
if (!filterContext.HttpContext.Request.IsAjaxRequest())
{
        filterContext.Result = new RedirectResult("~/Account/Login");
}
else
{
        //send error code to ajax request as session timeout
        filterContext.HttpContext.Response.StatusCode = 403;
        filterContext.HttpContext.Response.End();
```

```
}
```

g. Once this is done, we should check the status code i.e. 403 in the Ajax request call from the JavaScript. This can be achieved using following code.

```
$(document).ajaxError(
   function (e, request, settings, exception) {
      if (request.status == 403) {
         window.location.href = window.location.href
         return;
      }
 });
```

Note: Authentication and Authorization is an important topic and needs to be handled carefully to avoid any issues with the access rights. If anything is not clear, discuss with your Team Leader to seek any clarification.

Grid /List Controls

1. Every grid view or similar structure must implement Paging, Searching and Sorting based on business requirement.
2. Use of **Data Table Jquery plugin** is highly recommended.
3. Grid should have proper header, specifying the details Eg, Employee List, Product List etc..
4. Each column should have clear understandable Title.
5. It is preferred that all actions should be clubbed under single column with '|' separate links. Eg, Edit | View | Delete
6. Confirmation dialog box is mandatory for Delete operation.
7. **Paging/Pagination**
    a. Paging should be implemented at Database Level with the use of stored procedure.
    b. An option should be available for user to decide the page size, so option should be present with following values,
        i. 10
        ii. 20
        iii. 50
        iv. 100
        v. ALL
8. **Searching**
    a. Column level searching is recommended for implementing. However, decision has to be taken based on business requirement.
    b. Search should be enabled on significant columns.
    c. If possible and feasible, dropdown should be provided for search on columns like Status, Currency etc…
    d. If User searches on multiple columns then result will be based on AND operation.
9. **Sorting**
    a. By default data should be sorted according to primary key, unless and until specified otherwise.
    b. Sort icon should be present nearby to column header which will be indicative for user about sorting feature.
10. If there is no data available in data source, a "No Data available" message should be set as default.

11. Placeholder for grid control should be properly set, page should not flicker depending on different page sizes.
12. "Select All/None" functionality should be present in form of checkbox in the header row if there is a functionality of selection of records and performing certain action on selected records.
13. In case of admin side Grid View usage, an option to Export to Excel should be provided.
14. In case of Jquery Data table plugin, export to excel is available by default, so use of same is recommended wherever possible.

Modal Pop Ups

1. Only single pop up should be present on the page.
2. Modal Pop up box should be centrally aligned to screen.
3. Always Jquery based Modal pop up should be used.
4. Pop up should have a clearly visible (X) close button on right top corner.
5. Pop up should disappear on,
    a. Escape key.
    b. On click of any area within screen.
6. Pop up should have transparent overlay.
7. Cancel button should be available, if required.

Error/Exception Logging

1. In Asp.Net web applications (Web forms and MVC both), use of Elmah is mandatory.
2. Elmah configuration can be handled using following URL, https://www.asp.net/web-forms/overview/older-versions-getting-started/deploying-web-site-projects/logging-error-details-with-elmah-cs
3. Depending on business requirement, Log4Net or NLog API can be used to write file based (error) logs.
4. For Web Services/APIs error logging is mandatory.
5. Error Log should capture following details,
    a. Error/Exception message
    b. Inner exception
    c. Stack Trace
    d. Date Time
    e. Class/Function name etc..
6. All Logs should be descriptive and to be clearly written in adequate English.
7. Avoid using any abbreviations.

Error Pages

404, 500 error page should be handled through custom template

Handling system error codes like 404,500 errors, is important because the average site visitor is not aware what an HTTP 404 Error code means. Also, the default messages returned by webservers are too technical, poorly constructed and leave the visitors frustrated. Therefore it is the required to give your users something better and make your own custom 404 error page. Create different custom error pages as per error codes (404, 403, 500). You can set redirect using configuration in web.config file. That will give end user friendly message.

Example :

```
<system.web>
  <customErrors defaultRedirect="GenericError.htm"  mode="RemoteOnly">
   <error statusCode="500" redirect="InternalError.htm"/>
  </customErrors>
 </system.web>
```

Miscellaneous

1. For admin login page, CAPTCHA must be used
2. Breadcrumbs should be provided wherever necessary. (Optional : Depends on client)
3. For semantic and SEO reasons Site map should be provided wherever necessary. (Optional: Depends on client).